Syntactic parsing with NooJ

Max Silberztein
LASELDI, Université de Franche-Comté,
Besançon, France
max.silberztein@univ-fcomte.fr

ABSTRACT. *When parsing a text, NooJ's parsers store all the annotations that they produce in the Text's Annotation Structure (TAS). At each level of the various linguistic analyses and the corresponding parser, a given parser may add annotations to, or remove annotations from, the TAS. As annotations are attached to larger and larger sequences of texts, the TAS represents the hierarchical structure of the sentence and its syntactic constituents. We have added a new module that processes this structured information in order to display the structural tree of sentences. We discuss the difference between a structural tree and a derivation tree, and we show how NooJ's structural trees can represent any type of linguistic units, including discontinuous ones.*

1. Linguistic Units

One characteristic of NooJ is that its parsers process several types of linguistic units in texts: prefixes and suffixes (e.g. dis-, –ization), simple words (e.g. table), multi-word units (e.g. as a matter of fact) and discontinuous frozen expressions (e.g. to take … into account).[1] All linguistic units recognized by NooJ's morphological, lexical, syntactic and semantic parsers are represented as annotations, rather than tags.[2]

An annotation might represent either an Atomic Linguistic Unit (ALU), i.e. an element of the vocabulary of a language, or any type of sequences of ALUs that constitutes a meaningful syntactic or semantic unit, such as a noun phrase (e.g. the head of the company), a verbal group (e.g. may not have wanted to read) or an adverbial complement (e.g. Monday February the 11th at 2PM), etc.

When parsing a text, NooJ's parsers store all the annotations that they produce in the Text's Annotation Structure (TAS). Lexical annotations (ALUs) are displayed as

[1] Cf. (Silberztein 2008).

[2] Cf. (Silberztein 2007).

black labeled arrows in the TAS; syntactic annotations are generally displayed in green in the TAS. At each level of the various linguistic analyses and the corresponding parser (orthographical, lexical, morphological, syntactic and semantic), a given parser may add annotations to, or remove annotations from, the TAS. Usually parsers look up any type of information in the TAS, and enrich the TAS as they produce different types of analysis.

As a large number of linguistic units are ambiguous, the TAS often represents more than one annotation at a given location in the text, thus representing any type of ambiguity that might occur within one level of linguistic analysis, or between various levels of analysis. For instance, in the following figure that represents the TAS after the lexical parsing of a text, we can see that the multi-word unit round table (meaning a meeting) has been annotated in parallel with other literal interpretations for the same exact sequence. A number of ambiguities produced by low-level parsers might be solved by higher-level parsers: for instance it is easy to disambiguate the word "that" in the following sentence:

I brought that with that table.

When a syntactic parser produces an analysis for the whole sentence, it annotates the first occurrence as a pronoun and the second occurrence as a determiner.

Indeed, NooJ contains also a number of disambiguation mechanisms that can be used to remove annotations from the TAS, either automatically (via syntactic grammars selected in Info > Preferences), semi-automatically (via regular expressions selected from the TEXT > Locate panel) or manually (directly in the TAS window)[3].

However, it is often impossible to automatically disambiguate words, phrases or sentences with purely linguistic tools. For instance, in the sentence:

There is a round table in room A32

there is no way a computer (or even a human, for that matter) can detect for sure if the *round table* refers to a meeting or to a piece of furniture with a round shape. In the real world, figuring out what this *round table* refers to might involve a complex discourse analysis (to understand the semantic context of the sentence), or even a pragmatic analysis (to understand the context of the situation).

[3] see (Silberztein 2010a).

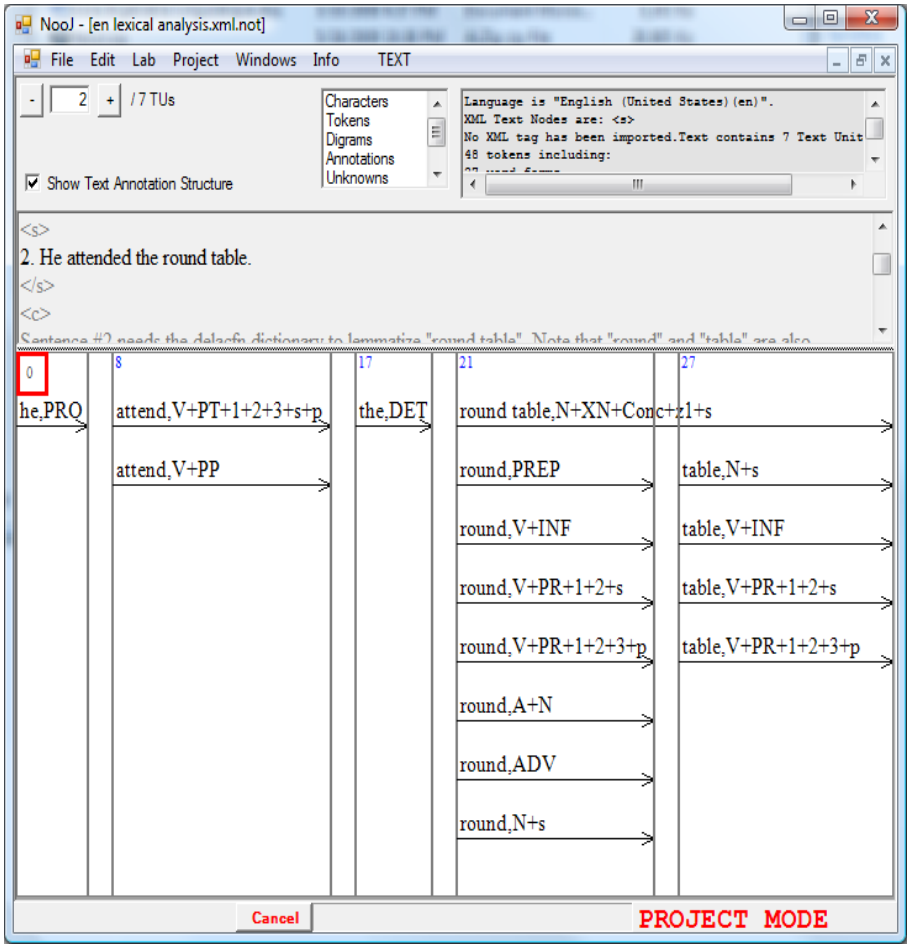

Figure 1. Lexical *ambiguity is represented by parallel paths*

In consequence, NooJ's TAS is designed to store potential ambiguities along several levels of linguistic analyses, and each of NooJ's parsers (at the morphology, syntactic and semantic levels) is designed to process ambiguous inputs and produce ambiguous outputs.

2. Structured Annotations

NooJ syntactic grammars are typically used in order to describe sequences of words that constitute meaningful units or entities (e.g. “the head of the European Central Bank”, “Monday, June the 12th at 3PM”). In other applications, linguists write syntactic grammars in order to locate syntactic constructions of interest, such as idiomatic or frozen expressions[4] or sentences that contain certain grammatical words or syntactic constructs.[5]

NooJ Syntactic grammars can associate matching (recognized) sequences in the text input with different kinds of information produced as an output; the information produced in the output of the syntactic grammars is represented as annotations that NooJ stores in the TAS. For instance, NooJ’s Date.nog syntactic grammar, which represents date complements such as “Monday, June 13th at a quarter to 9”, annotates every recognized date with the annotation <ADV+Date>; after having applied this grammar to a text, the corresponding annotations appear directly in the TAS in Figure 2.

A given grammar may produce more than one annotation for a given region of text in order to represent structural information for a phrase or a sentence. Moreover, as annotations can be recursively attached to substrings of the matching sequences, it is possible to represent and annotate embedded phrases.

[4] See for instance (Simonetta Vietri 2010).

[5] See for instance (Silberztein Max 2010b).

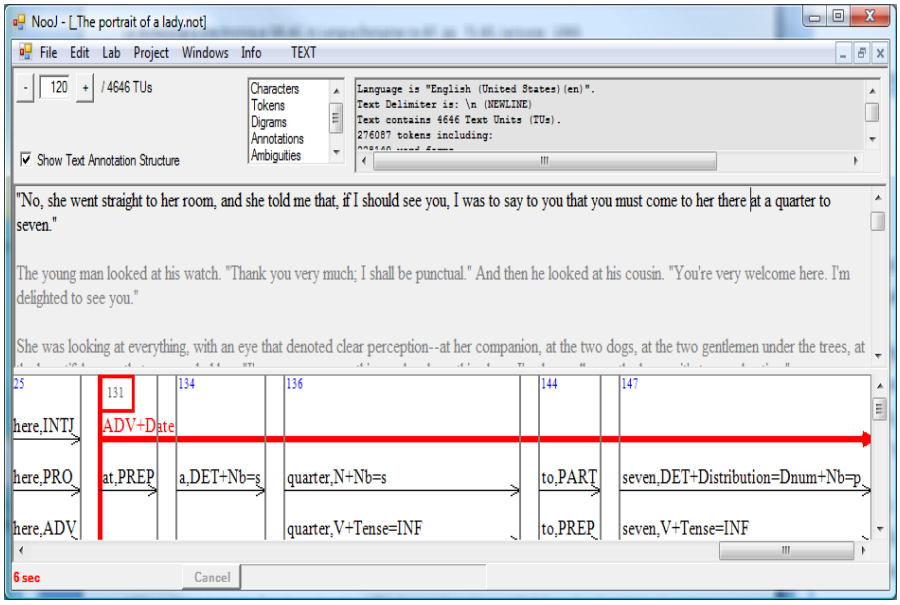

Figure 2. *A syntactic annotation in the TAS*

For instance, NooJ will annotate all the sentences recognized by the syntactic grammar of Figure 3 as <SENTENCE>; inside each of these annotated sentences, the sequence of words recognized by the embedded graph NP at the left of the verb will be annotated as <NP+Sub>, the sequence recognized by the graph VG will be annotated as <VG>, and the sequence recognized by the graph NP after the verb will be annotated as <NP+Obj>. Inside each of the graphs NP and VG (which themselves contain other embedded graphs), there might or might not be other annotations.

When such grammars are applied to a given text, the resulting TAS contains annotations that are structured, and the structure of the TAS reflects on the structure of the original sentence or phrase in Figure 4.

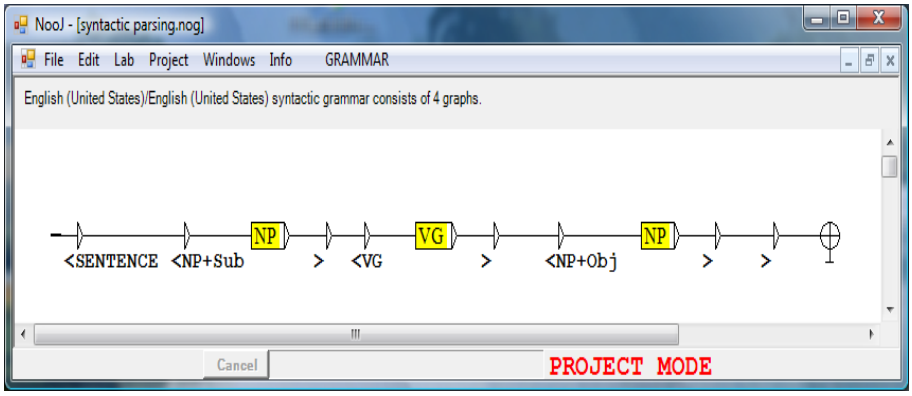

Figure 3. A *grammar may produce embedded syntactic annotations*

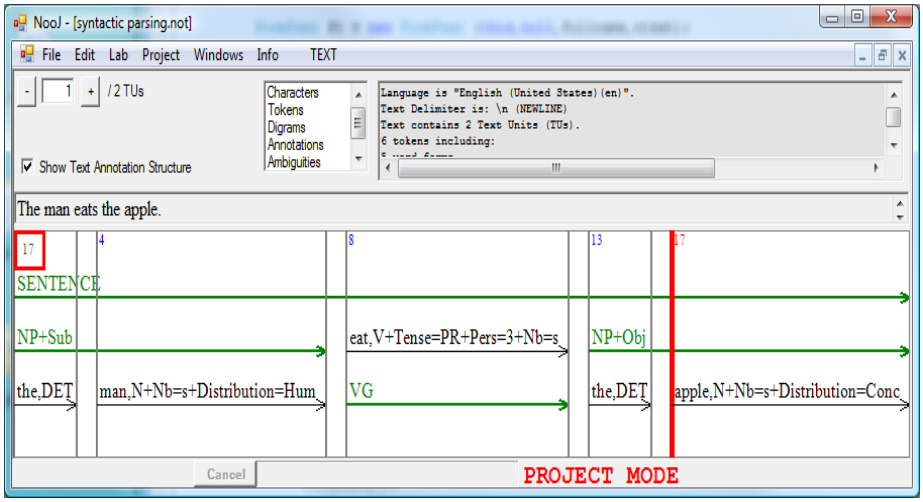

Figure 4. A *structured system of annotations*

It is important to realize that the syntactic TAS produced after a syntactic analysis has a different nature than the lexical TAS:

-- in the lexical TAS, parallel annotations correspond to various types of ambiguities. For instance, in Figure 1, the annotation for “round table” is parallel to the sequence of two annotations for “round” followed by “table”. Obviously, these two analyses are concurrent, as they cannot exist at the same time.

-- in the syntactic TAS, parallel annotations may correspond to different levels of constituents in the phrase or sentence structure. For instance, in Figure 4, the annotation <SENTENCE> co-exists with the sequence of the three annotations <NP+Sub>, <VG> and <NP+Obj>. Of course, these parallel paths are both valid at the same time: they correspond to two different levels in the structure.[6]

We have added the functionality of displaying this structured set of annotations to NooJ in the form of a structural tree that represents the structure of the recognized sequence, usually a phrase or a sentence (CONCORDANCE > Display Syntactic Analysis), see Figure 5:

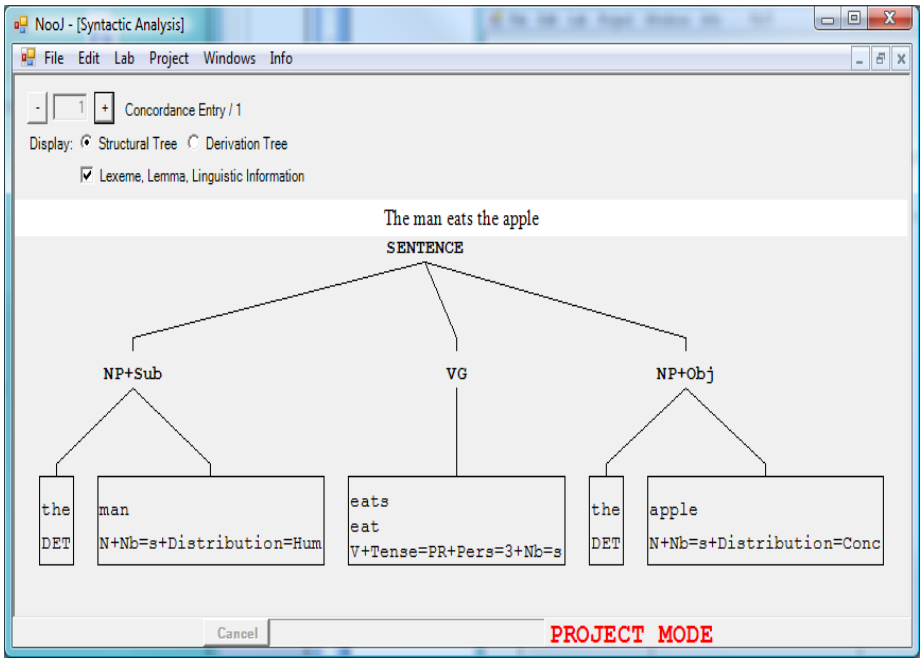

Figure 5. The *corresponding structural tree*

[6] Note however that some lexical ambiguities might still remain in the TAS after the syntactic parsing.

Note finally that if there are more than one set of syntactic annotations, or if there are remaining lexical ambiguities in the TAS, NooJ will produce and display more than one syntactic structural tree.

3. Structural trees *vs* Derivation trees

NooJ can display both structural and derivation trees and it is important to understand their difference. NooJ's structural trees, such as the one in Figure 5 that was produced by the grammar of Figure 3, have a different nature than derivation trees produced by traditional syntactic parsers.

NooJ structural trees represent the structure of the annotations that were attached to the recognized sentences or phrases, whereas derivation trees represent the syntactic parsing process, i.e. the path in the grammar that the parser took while it was parsing the text.

For instance, consider the following grammar that contains a number of embedded rules (or graphs):

```
NP = :Det N :Modif ;
Det = :DefinedDet | :UndefinedDet | :NominalDet |
:AdjectivalDet ;
DefinedDet = :Article | :PossessiveDet |
:DemonstrativeDet ;
Article = the;
PossessiveDet = my | your | his | her | our | their;
DemonstrativeDet = this | that | these | those;
```

…

The corresponding derivation tree is displayed in Figure 6. Traditional syntactic parsers produce derivation trees as their main result. Although derivation trees are very useful for figuring out how a given text was actually parsed[7], they represent the structure of the grammars, rather than the structure of the sentence. And, of course, derivation trees are highly dependent on the structure of the grammar: if a linguist decides to reorganize the DET grammar for instance, the derivation trees produced by the parser will change accordingly.

[7] NooJ's grammar debugger uses derivation trees to display the path of each matching solution (see Grammar > Debugger).

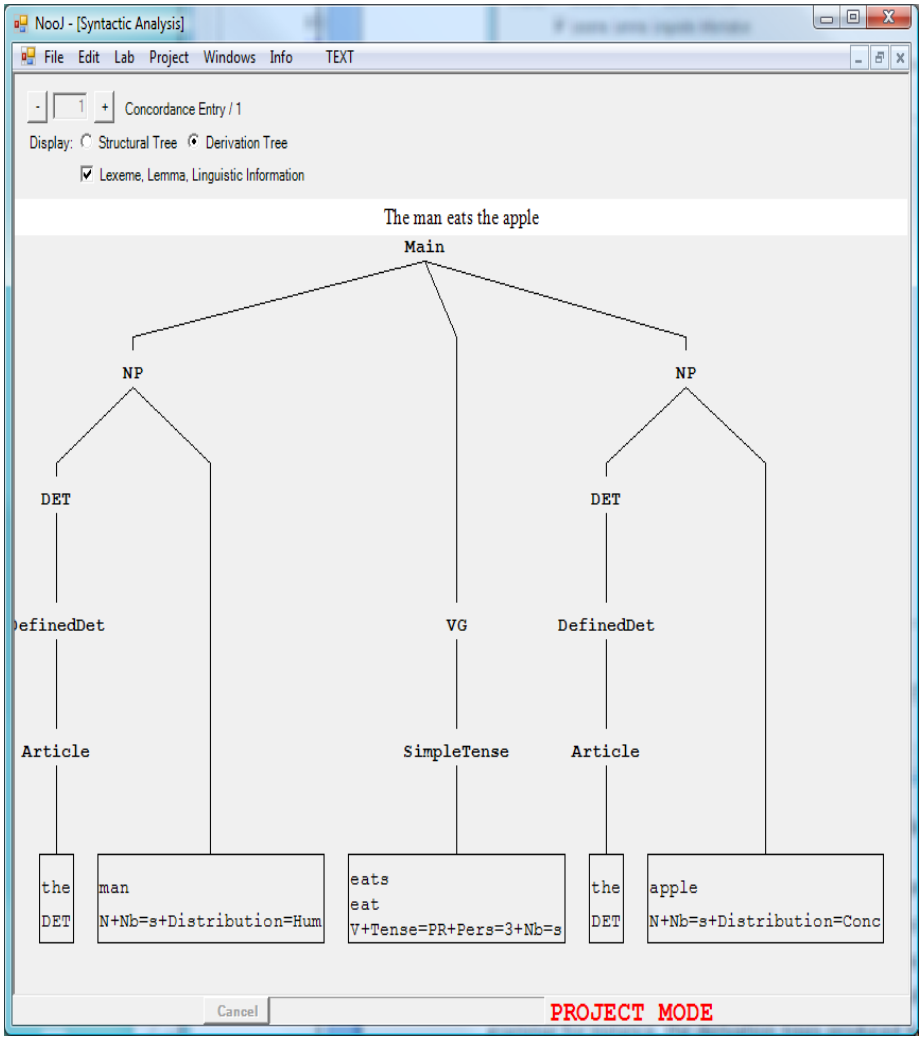

Figure 6. *The corresponding derivation tree*

On the contrary, the structural information represented in NooJ's structural trees is quite independent from the way the grammar is organized: that independence allows different linguists to construct and accumulate a large number of grammars over time, and their merging does not jeopardize the integrity of the resulting analysis.

4. Complex Atomic Linguistic Units

NooJ's syntactic parser processes represents affixes, simple words, multi-word units as well as discontinuous expressions in a unified way.

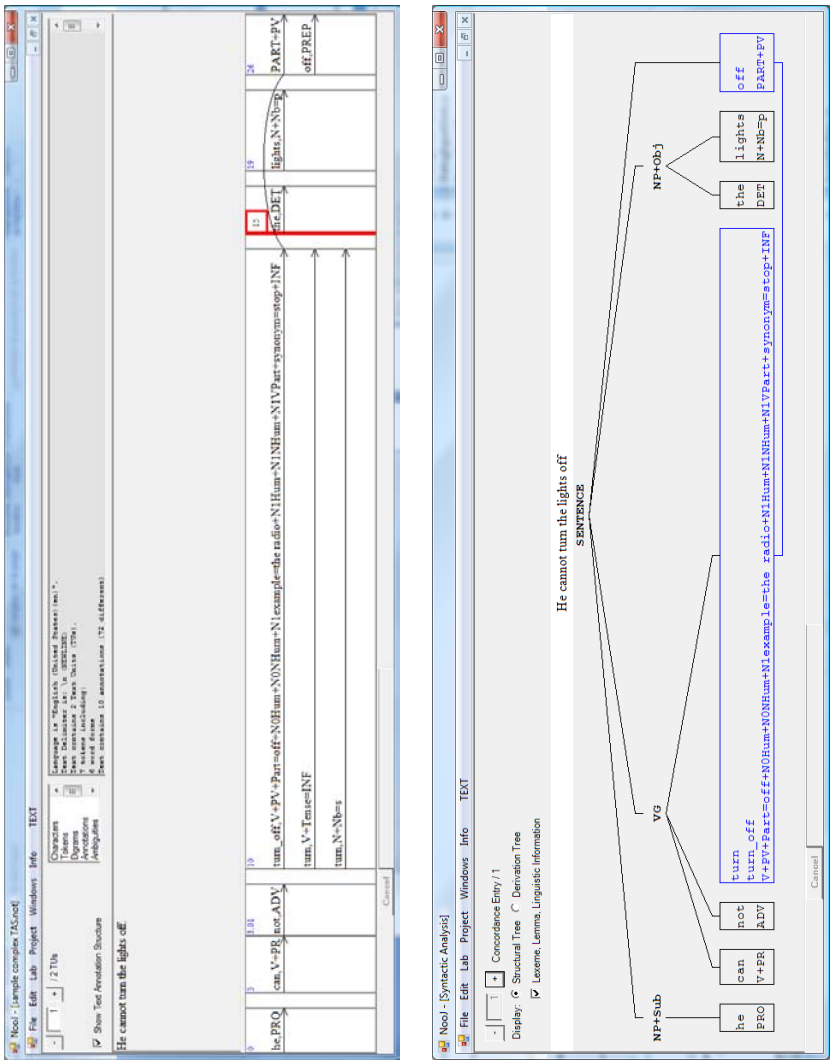

Figure 7.*A complex TAS and the corresponding syntactic tree*

Figure 7 shows the TAS for the text "He cannot turn the lights off": the contracted word cannot is represented by two annotations, whereas the discontinuous phrasal verb[8] turn off is represented by a "bridge-type" annotation:

Since NooJ's syntactic parser operates on the TAS rather than on the text itself, it processes the two annotations "<can,V>" and "<not,ADV>" as if they were two separate words. The discontinuous annotation for the phrasal verb "<turn off,V+PV>" is not a problem for NooJ's syntactic parser either: NooJ knows how to handle discontinuous annotations.

The ability to process all types of ALUs, including contracted or agglutinated as well as simple words, allows linguists to design simple syntactic grammars. For instance, in French, the word form *au* is a contraction of the preposition *à* and the determiner *le*. Usually, these two components do not belong to the same phrase: the preposition *à* belongs to the main verb's structure (e.g. *parler à quelqu'un*) or to a prepositional phrase (e.g. *être à Paris*), whereas determiners introduce noun phrases. Because NooJ transparently solves contractions, syntactic grammars built into NooJ do not need to take them into account: a regular, simple grammar for noun phrases that start with a determiner will automatically work in NooJ, even if the initial determiner happens to be contracted.

Figure 8 shows that the contracted word aux is processed as two ALUs (à and les) that belong to different substructures of the structural tree. This capability is useful for taking care of the dozen contractions in English and in Romance languages, and it is absolutely crucial for agglutinative languages such as Arabic and Hebrew, where series of grammatical words that belong to different structures of the sentence are agglutinated: in these languages, sequences such as "and in my house" are written as one word form[9].

[8] See (Machonis 2008).

[9] See (Mesfar 2010).

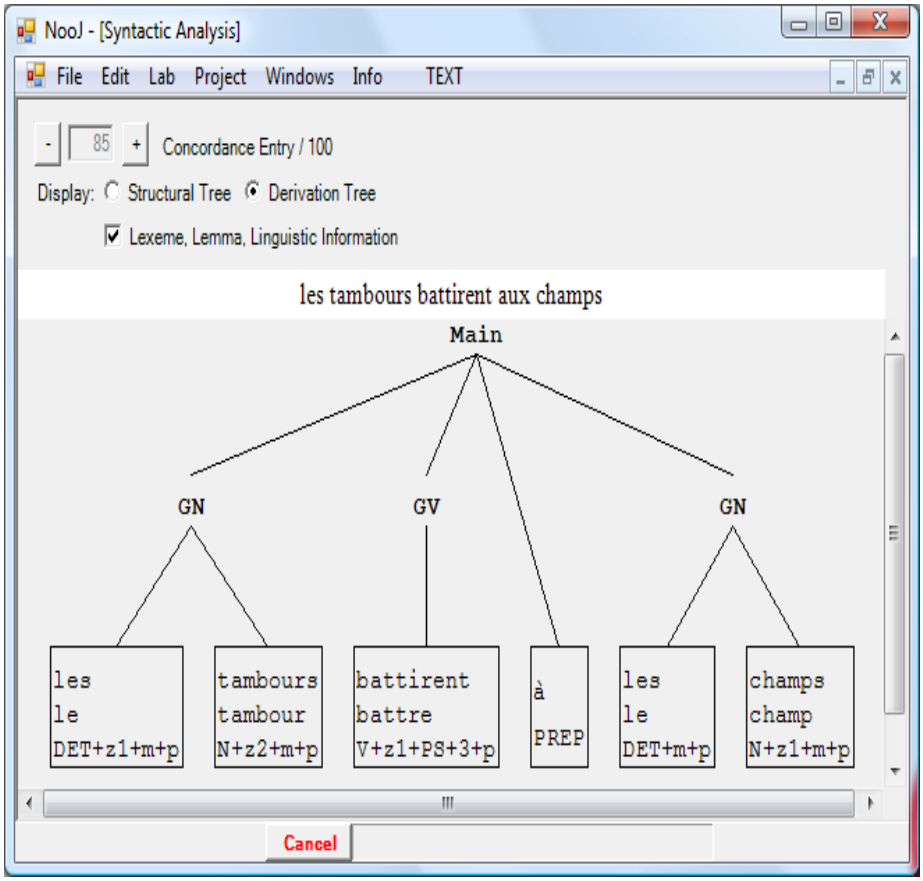

Figure 8. *Syntactic parsers process contracted words transparently*

5. Disambiguation

Until now, the only method for automatically disambiguating words in texts was to filter out annotations from the TAS by applying local grammars. Although local grammars are easy to develop, perform very efficiently, and produce spectacular

results, in most cases, only a syntactic analysis of the whole sentence can produce a perfect syntactic disambiguation of all ALUs in the sentence.[10]
NooJ's syntactic parser makes it easy to perform this perfect syntactic disambiguation: when one applies a syntactic grammar to the text with the option "syntactic analysis" checked, NooJ keeps all matching ALUs in the concordance; the concordance entry can then be reapplied to the original TAS as a filter, which in effect removes all other ALUs: the resulting TAS is disambiguated. Figure 4 shows such a resulting TAS: all the ALUs for the verb "to man" (such as in "you man the ship").

6. Conclusion

NooJ's syntactic parser now allows linguists to build and accumulate larger syntactic grammars up to the sentence level. As opposed to traditional syntactic parsers, NooJ processes and represents all types of linguistic units: affixes, simple words, multi-word units and also discontinuous expressions, which are transparently represented as chains of linked leaves in the syntactic tree.

NooJ, as opposed to traditional syntactic parsers, distinguishes structural trees from derivational trees: in particular, structural trees represent the structure of the text which may be largely independent and different from the structure of the grammar that was used to parse the text. This allows structural trees to be much more robust and standardized, as any local or global reorganization of a syntactic grammar does not need to have any impact on the representation of the resulting sentence structure.

This framework allows NooJ to process contractions and agglutinations transparently: for instance, contracted words have no impact in the design of syntactic grammar because they are taken care of *before* the syntactic parsing. Another characteristic of NooJ's syntactic parser is that it does not require the text's linguistic units to be fully disambiguated beforehand: in particular, lexical ambiguities between multi-word units and sequences of simple words may remain unresolved, both before and after the syntactic parsing of texts.

Structural syntactic grammars are used to produce a structural analysis of the matching phrases and sentences, which can then be displayed as trees. As opposed

[10] By perfect syntactic disambiguation, I mean that all the ALUs that cannot possibly occur in the sentence, according to syntactic rules, will be deleted. Of course, the TAS still might contains a number of remaining lexical ambiguities, see the ambiguity in "there is a round table in room A32" (a meeting or a table?).

to traditional syntactic parsers, NooJ allows linguists to control the structure produced independently from the organization of the grammar. I believe this feature is crucial to the project of accumulating and sharing large numbers of grammars.
Finally, the resulting structured matches can be used as filters to disambiguate the TAS in a more powerful and systematic way than possible when using local grammars.

References

Machonis Peter, 2008. NooJ: a practical method for Parsing Phrasal Verbs. In Proceedings of the 2007 International NooJ Conference. Cambridge Scholars Publishing: Newcastle (pp 149-161).

Mesfar Slim, 2010. Morphological Grammars for Standard Arabic Tokenization. In Proceedings of the 2008 International NooJ Conference. Cambridge Scholars Publishing: Newcastle (pp 114-127).

Silberztein Max, 2007. An Alternative Approach to Tagging. Invited Paper In Proceedings of NLDB 2007. LNCS series, Springer-Verlag Eds (pp. 1-11).

Silberztein Max, 2008. Complex Annotations with NooJ. In Proceedings of the 2007 International NooJ Conference. Cambridge Scholars Publishing: Newcastle (pp 214-227).

Silberztein Max, 2010a. Disambiguation Tools for NooJ. In Proceedings of the 2008 International NooJ Conference. Cambridge Scholars Publishing: Newcastle (pp. 158-171).

Silberztein Max, 2010b. Analyse automatique des locutions en *sous* et *sur*. In Le français moderne, July 2010.

Vietri Simonetta, 2010. The Formalization of Italian Lexicon-Grammar Tables in a NooJ pair Dictionary/Grammar. In Proceedings of the 2008 International NooJ Conference. Cambridge Scholars Publishing: Newcastle (pp. 182-191).